A Conditional Logical Framework

Luigi Liquori, Furio Honsell, Marina Lenisa, Ivan Scagnetto

A Conditional Logical Framework ⋆

Furio Honsell, Marina Lenisa, Luigi Liquori, and Ivan Scagnetto

INRIA, France & UNIUD, Italy
[honsell,lenisa,scagnett]@dimi.uniud.it,Luigi.Liquori@inria.fr

Abstract. The *Conditional Logical Framework* LF_κ is a variant of the Harper-Honsell-Plotkin's Edinburgh Logical Framemork LF. It features a generalized form of λ-abstraction where β-reductions fire *under the condition* that the argument satisfies a *logical predicate*. The key idea is that the type system *memorizes* under what conditions and where reductions have yet to fire. Different notions of β-reductions corresponding to different predicates can be combined in LF_κ. The framework LF_κ subsumes, by simple instantiation, LF (in fact, it is also a subsystem of $\mathsf{LF}!$), as well as a large class of new generalized conditional λ-calculi. These are appropriate to deal smoothly with the side-conditions of both Hilbert and Natural Deduction presentations of Modal Logics. We investigate and characterize the metatheoretical properties of the calculus underpinning LF_κ, such as subject reduction, confluence, strong normalization.

1 Introduction

The Edinburgh Logical Framework LF of [HHP93] was introduced both as a general theory of logics and as a metalanguage for a generic proof development environment. In this paper, we consider a variant of LF, called *Conditional Logical Framework* LF_κ, which allows to deal uniformly with logics featuring *side-conditions* on the application of inference rules, such as *Modal Logics*. We study the language theory of LF_κ and we provide proofs for subject reduction, confluence, and strong normalization. By way of example, we illustrate how special instances of LF_κ allow for smooth encodings of Modal Logics both in Hilbert and Natural Deduction style.

The motivation for introducing LF_κ is that the type system of LF is too coarse as to the "side conditions" that it can enforce on the application of rules. Rules being encoded as functions from proofs to proofs and rule application simply encoded as lambda application, there are only roundabout ways to encode provisos, even as simple as that appearing in a *rule of proof*. Recall that a rule of proof can be applied only to premises which do not depend on any assumption, as opposed to a *rule of derivation* which can be applied everywhere. Also rules which appear in many natural deduction presentations of Modal and Program Logics are very problematic in standard LF. Many such systems feature rules which can be applied only to premises which depend solely on assumptions of a particular shape [CH84], or whose derivation has been carried out using only certain sequences of rules. In general, Modal, Program, Linear or Relevance

⋆ Supported by AEOLUS FP6-IST-FET Proactive.

Logics appear to be encodable in LF only encoding a very heavy machinery, which completely rules out any natural Curry-Howard paradigm, see *e.g.* [AHMP98]. As we will see for Modal Logics, LF_κ allows for much simpler encodings of such rules, which open up promising generalizations of the proposition-as-types paradigm.

The idea underlying the Conditional Logical Framework LF_κ is inspired by the Honsell-Lenisa-Liquori's *General Logical Framework* GLF see [HLL07], where we proposed a uniform methodology for extending LF, which allows to deal with pattern matching and restricted λ-calculi. The key idea, there, is to separate two different notions that are conflated in the original LF. As already mentioned, much of the rigidity of LF arised from the fact that β-reduction can be applied always in full generality. One would like to fire a β-reduction under certain conditions on typed terms, but the type system is not rich enough to be able to express such restrictions smoothly. What we proposed in [HLL07] is to use as type of an application, in the term application rule, $(\mathsf{O}{\cdot}\mathsf{Appl})$ below, not the type which is obtained by carrying out directly in the metalanguage the substitution of the argument in the type, but a new form of type which simply records the information that such a reduction can be carried out. An application of the Type Conversion Rule can then recover, if possible, the usual effect of the application rule. This key idea leads to the following object application rule:

$$\frac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : (\lambda x{:}A.B)\,N}$$

Once this move has been made, we have a means of annotating in a type the information that a reduction *is waiting to be carried out in the term*. If we take seriously this move, such a type need not be necessarily definitionally equal to the reduced one as in the case of LF. Without much hassle we have a principled and natural way of typing calculi featuring generalized or restricted forms of β-reduction, which *wait for some condition to be satisfied before they can fire*. Furthermore, such calculi can be used for underpinning new powerful Logical Frameworks, where all the extra complexity in terms can be naturally tamed using the expressive power of the new typing system. Once this program is carried out in a sufficiently modular form, we have a full-fledged Logical Framework.

More specifically, in LF_κ we consider a new form of λ and corresponding Π abstraction, *i.e.* $\lambda_{\mathcal{P}} x{:}A.M$ and $\Pi_{\mathcal{P}} x{:}A.M$, where $\mathcal{P}$ is a predicate, which ranges over a suitable set of predicates. The reduction $(\lambda_{\mathcal{P}} x{:}A.M)\,N$ fires only if the predicate $\mathcal{P}$ holds on N, and in this case the redex progresses, as usual, to $M[N/x]$. Therefore the final object application rule in LF_κ will be:

$$\frac{\Gamma \vdash_\Sigma M : \Pi_{\mathcal{P}} x{:}A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma M\,N : (\lambda_{\mathcal{P}} x{:}A.B)\,N}$$

In this rule a type where a reduction is "stuck", if the predicate $\mathcal{P}$ is not true on N, is assigned to an object application. However, when we view this object as a subterm of another term, such reduction could become allowed in the future, after other reductions are performed in the term, which provide substitutions for N. In LF_κ more predicates can be combined. LF_κ subsumes standard LF, which is recovered by considering the trivial predicate that is constantly true.

Historically, the idea of introducing stuck-reduction in objects and types, in the setting of higher-order term rewriting systems with sophisticated pattern-matching capabilities, was first introduced in Cirstea-Kirchner-Liquori's Rho-cube [CKL01b], in order to design a hierarchy of type systems for the untyped Rewriting Calculus of [CKL01a], and then it was generalized to a more general framework of Pure Type

Systems with Patterns [BCKL03]. This typing protocol was essential to preserve the strong normalization of typable terms, as proved in [HLL07]. The idea underlying the Conditional Logical Framework LF_κ is the same exploited in [HLL07] for the General Logical Framework GLF. However, there is an important difference between the two frameworks in the definition of predicates. On one hand, predicates in [HLL07] are used both to determine whether β-reduction fires and to compute a substitution, while in the present paper they are used only to determine whether β-reduction fires. On the other hand, in [HLL07] predicates are defined on terms, while here they are defined on typed judgments. This adds extra complexity both in the definition of the system and in the study of its properties, but it greatly simplifies the treatment of Modal Logics and of other situations where conditions depending on types have to be expressed.

Apart from Modal Logics, we believe that our Conditional Logical Framework could also be very helpful in modeling dynamic and reactive systems: for example bio-inspired systems, where reactions of chemical processes take place only provided some extra structural or temporal conditions; or process algebras, where often no assumptions can be made about messages exchanged through the communication channels. Indeed, it could be the case that a redex, depending on the result of a communication, can remain stuck until a "good" message arrives from a given channel, firing in that case an appropriate reduction (this is a common situation in many protocols, where "bad" requests are ignored and "good ones" are served). Such dynamical (run-time) behaviour could be hardly captured by a rigid type discipline, where bad terms and hypotheses are ruled out *a priori*, see *e.g.* [NPP08].

In this paper we develop all the metatheory of LF_κ. In particular, we prove subject reduction, strong normalization, confluence; this latter under the sole assumption that the various predicate reductions nicely combine, *i.e.* no reduction can prevent a redex, which could fire, from firing after the reduction. Since β-reduction in LF_κ is defined only on typed terms, in order to prove subject reduction and confluence, we need to devise a new approach, alternative to the one in [HHP93]. Our approach is quite general, and in particular it yields alternative proofs for the original LF.

In conclusion, the work on LF_κ carried out in this paper is valuable in three ways. First, being LF_κ so general, the results in this paper potentially apply to a wide range of Logical Frameworks, therefore many fundamental results are proved only once and uniformly for all systems. Secondly, the LF_κ approach is useful in view of implementing a "telescope" of systems, since it provides relatively simple sufficient conditions to test whether a potential extension of the framework is safe. Thirdly, LF_κ can suggest appropriate extensions of the proposition-as-types paradigm to a wider class of logics.

Synopsis. In Section 2, we present the syntax of LF_κ, its type system, and the predicate reduction. In Section 3, we present instantiations of LF_κ to known as well as to new calculi, and we show how to encode smoothly Modal Logics. The LF_κ's metatheory is carried out in Section 4. Conclusions and directions for future work appear in Section 5. Proofs appear in a Web Appendix available at the author's web pages.

2 The System

Syntax. In the following definition, we introduce the LF_κ pseudo-syntax for kinds, families, objects, signatures and contexts.

Definition 1 (LF_κ Pseudo-syntax)

$\Sigma \in \mathcal{S}$	$\Sigma ::= \emptyset \mid \Sigma, a{:}K \mid \Sigma, f{:}A$	*Signatures*
$\Gamma, \Delta \in \mathcal{C}$	$\Gamma ::= \emptyset \mid \Gamma, x{:}A$	*Contexts*
$K \in \mathcal{K}$	$K ::= \mathsf{Type} \mid \Pi_{\mathcal{P}} x{:}A.K \mid \lambda_{\mathcal{P}} x{:}A.K \mid K\,M$	*Kinds*
$A, B, C \in \mathcal{F}$	$A ::= a \mid \Pi_{\mathcal{P}} x{:}A.B \mid \lambda_{\mathcal{P}} x{:}A.B \mid A\,M$	*Families*
$M, N, Q \in \mathcal{O}$	$M ::= f \mid x \mid \lambda_{\mathcal{P}} x{:}A.M \mid M\,N$	*Objects*

where a, f are typed constants standing for fixed families and terms, respectively, and $\mathcal{P}$ is a predicate ranging over a set of predicates, which will be specified below.

LF_κ is parametric over a set of predicates of a suitable shape. Such predicates are defined on typing judgments, and will be discussed in the section introducing the type system.

Notational conventions and auxiliary definitions. Let "T" range over any term in the calculus (kind, family, object). The abstractions $\checkmark_{\mathcal{P}} x{:}A.T$ ($\checkmark \in \{\lambda, \Pi\}$) bind the variable x in T. Domain $\mathsf{Dom}(\Gamma)$ and codomain $\mathsf{CoDom}(\Gamma)$ are defined as usual. Free $\mathsf{Fv}(T)$ and bound $\mathsf{Bv}(T)$ variables are defined as usual. As usual, we suppose that, in the context $\Gamma, x{:}T$, the variable x does not occur free in Γ and T. We work modulo α-conversion and Barendregt's hygiene condition.

Type System. LF_κ involves type judgments of the following shape:

$\Sigma\ \mathsf{sig}$	Σ is a valid signature
$\vdash_\Sigma \Gamma$	Γ is a valid context in Σ
$\Gamma \vdash_\Sigma K$	K is a kind in Γ and Σ
$\Gamma \vdash_\Sigma A : K$	A has kind K in Γ and Σ
$\Gamma \vdash_\Sigma M : A$	M has type A in Γ and Σ
$\Gamma \vdash_\Sigma T \mapsto_\beta T' (: T'')$	T reduces to T' in Γ, Σ (and T'')
$\Gamma \vdash_\Sigma T =_\beta T' (: T'')$	T converts to T' in Γ, Σ (and T'')

The typing rules of LF_κ are presented in Figure 1. As remarked in the introduction, rules ($\mathsf{F{\cdot}Appl}$) and ($\mathsf{O{\cdot}Appl}$) do not utilize metasubstitution as in standard LF, but rather introduce an explicit type redex. Rules ($\mathsf{F{\cdot}Conv}$) and ($\mathsf{O{\cdot}Conv}$) allow to recover the usual rules, if the reduction fires.

Typed Operational Semantics. The "type driven" operational semantics is presented in Figure 2, where the most important rule is ($\mathsf{O{\cdot}Red}$), the remaining ones being the contextual closure of β-reduction. For lack of space we omit similar rules for kinds and constructors. According to rule ($\mathsf{O{\cdot}Red}$), reduction is allowed only if the argument in the context satisfies the predicate $\mathcal{P}$. In this sense, reduction becomes "conditioned" by $\mathcal{P}$. In LF_κ, we can combine more predicate reductions, *i.e.*, we can define and combine *several predicates* guarding β-reduction, whose shape is as follows. Each predicate is determined by a set $\mathcal{A}$ of families (types), and the intended meaning is that it holds on

Signature and Context rules

$$\frac{}{\emptyset \text{ sig}}\ (\text{S·Empty}) \qquad \frac{\Sigma \text{ sig}}{\vdash_\Sigma \emptyset}\ (\text{C·Empty})$$

$$\frac{\Sigma \text{ sig} \quad \vdash_\Sigma K \quad a \notin \mathsf{Dom}(\Sigma)}{\Sigma, a{:}K \text{ sig}}\ (\text{S·Kind})$$

$$\frac{\Sigma \text{ sig} \quad \vdash_\Sigma A : \mathsf{Type} \quad f \notin \mathsf{Dom}(\Sigma)}{\Sigma, f{:}A \text{ sig}}\ (\text{S·Type})$$

$$\frac{\vdash_\Sigma \Gamma \quad \Gamma \vdash_\Sigma A : \mathsf{Type} \quad x \notin \mathsf{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x{:}A}\ (\text{C·Type})$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B : \mathsf{Type}}{\Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.B : \mathsf{Type}}\ (\text{F·Pi})$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.B : \Pi_\mathcal{P} x{:}A.K}\ (\text{F·Abs})$$

$$\frac{\Gamma \vdash_\Sigma A : \Pi_\mathcal{P} x{:}B.K \quad \Gamma \vdash_\Sigma N : B}{\Gamma \vdash_\Sigma A\, N : (\lambda_\mathcal{P} x{:}B.K)\, N}\ (\text{F·Appl})$$

$$\frac{\Gamma \vdash_\Sigma A : K' \quad \Gamma \vdash_\Sigma K \quad \Gamma \vdash_\Sigma K =_\beta K'}{\Gamma \vdash_\Sigma A : K}\ (\text{F·Conv})$$

Kind rules

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma \mathsf{Type}}\ (\text{K·Type})$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.K}\ (\text{K·Pi})$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.K}\ (\text{K·Abs})$$

$$\frac{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.K \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.K)\, N}\ (\text{K·Appl})$$

Family rules

$$\frac{\vdash_\Sigma \Gamma \quad a{:}K \in \Sigma}{\Gamma \vdash_\Sigma a : K}\ (\text{F·Const})$$

Object rules

$$\frac{\vdash_\Sigma \Gamma \quad x{:}A \in \Gamma}{\Gamma \vdash_\Sigma x : A}\ (\text{O·Var})$$

$$\frac{\vdash_\Sigma \Gamma \quad f{:}A \in \Sigma}{\Gamma \vdash_\Sigma f : A}\ (\text{O·Const})$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M : \Pi_\mathcal{P} x{:}A.B}\ (\text{O·Abs})$$

$$\frac{\Gamma \vdash_\Sigma M : \Pi_\mathcal{P} x{:}A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma M\, N : (\lambda_\mathcal{P} x{:}A.B)\, N}\ (\text{O·Appl})$$

$$\frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma B : \mathsf{Type} \quad \Gamma \vdash_\Sigma A =_\beta B : \mathsf{Type}}{\Gamma \vdash_\Sigma M : B}\ (\text{O·Conv})$$

Fig. 1. LF_K Type System

a typed judgment $\Gamma \vdash_\Sigma M : A$ and a set of variables $\mathcal{X} \subseteq \mathsf{Dom}(\Gamma)$ if *"$\Gamma \vdash_\Sigma M : A$ is derivable and all the free variables in M which are in $\mathcal{X}$ appear in subterms typable with a type in $\mathcal{A}$"*. This intuition is formally stated in the next definition.

Definition 2 (Good families (types) and predicates)
Let $\mathcal{A} \subseteq \mathcal{F}$ be a set of families. This induces a predicate $\mathcal{P}_\mathcal{A}$ (denoted by $\mathcal{P}$, for simplicitly), defined on typed judgments $\Gamma \vdash M : A$ and sets $\mathcal{X}$ such that $\mathcal{X} \subseteq \mathsf{Dom}(\Gamma)$. The truth table of $\mathcal{P}$ appears in Figure 3.
We call good *a predicate $\mathcal{P}$ defined as above, and* good types *the set of types in $\mathcal{A}$ inducing it.*

The following lemma states formally the intended meaning of our predicates:

Lemma 1 ($\mathcal{P}$ Satisfiability).
Given a predicate $\mathcal{P} \in \mathcal{L}$ induced by a set of families (types) $\mathcal{A}$, $\mathcal{P}$ holds on a typed

$$\frac{\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.M)\, N : C \quad \Gamma \vdash_\Sigma M[N/x] : C \qquad \mathcal{P}(\mathsf{Fv}(N); \Gamma \vdash_\Sigma N : A)}{\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.M)\, N \mapsto_\beta M[N/x] : C}\ (\mathsf{O{\cdot}Red})$$

$$\frac{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M : \Pi_\mathcal{P} x{:}A.B \quad \Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.N : \Pi_\mathcal{P} x{:}A.B \quad \Gamma, x{:}A \vdash_\Sigma M \mapsto_\beta N : B}{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M \mapsto_\beta \lambda_\mathcal{P} x{:}A.N : \Pi_\mathcal{P} x{:}A.B}\ (\mathsf{O{\cdot}\lambda{\cdot}Red_1})$$

$$\frac{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M : C \quad \Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}B.M : C \quad \Gamma \vdash_\Sigma A \mapsto_\beta B : \mathsf{Type}}{\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M \mapsto_\beta \lambda_\mathcal{P} x{:}B.M : C}\ (\mathsf{O{\cdot}\lambda{\cdot}Red_2})$$

$$\frac{\Gamma \vdash_\Sigma M\, N : (\lambda_\mathcal{P} x{:}A.B)\, N \quad \Gamma \vdash_\Sigma P\, N : (\lambda_\mathcal{P} x{:}A.B)\, N \quad \Gamma \vdash_\Sigma M \mapsto_\beta P : \Pi_\mathcal{P} x{:}A.B}{\Gamma \vdash_\Sigma M\, N \mapsto_\beta P\, N : (\lambda_\mathcal{P} x{:}A.B)\, N}\ (\mathsf{O{\cdot}Appl{\cdot}Red_1})$$

$$\frac{\Gamma \vdash_\Sigma M\, N : (\lambda_\mathcal{P} x{:}A.B)\, N \quad \Gamma \vdash_\Sigma M\, P : (\lambda_\mathcal{P} x{:}A.B)\, N \quad \Gamma \vdash_\Sigma N \mapsto_\beta P : A}{\Gamma \vdash_\Sigma M\, N \mapsto_\beta M\, P : (\lambda_\mathcal{P} x{:}A.B)\, N}\ (\mathsf{O{\cdot}Appl{\cdot}Red_2})$$

$$\frac{\Gamma \vdash_\Sigma M \mapsto_\beta N : A \quad \Gamma \vdash_\Sigma A =_\beta B : \mathsf{Type}}{\Gamma \vdash_\Sigma M \mapsto_\beta N : B}\ (\mathsf{O{\cdot}Conv{\cdot}Red})$$

Fig. 2. LF_K Reduction (Object rules)

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad A \in \mathcal{A}}{\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M : A)}\ (\mathsf{O{\cdot}Start_1}) \qquad \frac{\Gamma \vdash_\Sigma M{:}A}{\mathcal{P}(\emptyset; \Gamma \vdash_\Sigma M : A)}\ (\mathsf{O{\cdot}Start_2})$$

$$\frac{\mathcal{P}(\mathcal{X}; \Gamma, x{:}A \vdash_\Sigma M : B)}{\mathcal{P}(\mathcal{X} \setminus \{x\}; \Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.M : \Pi_\mathcal{P} x{:}A.B)}\ (\mathsf{O{\cdot}Abs})$$

$$\frac{\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M : \Pi_\mathcal{P} x{:}A.B) \quad \mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma N : A)}{\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M\, N : (\lambda_\mathcal{P} x{:}A.B)\, N)}\ (\mathsf{O{\cdot}Appl})$$

$$\frac{\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M : A) \quad \Gamma \vdash_\Sigma B : \mathsf{Type} \quad \Gamma \vdash_\Sigma A =_\beta B : \mathsf{Type}}{\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M : B)}\ (\mathsf{O{\cdot}Conv})$$

Fig. 3. $\mathcal{P}$'s truth table

judgment $\Gamma \vdash_\Sigma M : B$ and a set of variables $\mathcal{X} \subseteq \mathsf{Dom}(\Gamma)$, if $\Gamma \vdash_\Sigma M : B$ is derivable and all the free variables in M which are in $\mathcal{X}$ appear in subterms typable with a type in $\mathcal{A}$.

Hence, if we take $\mathcal{X} = \mathsf{Fv}(M)$, then $\mathcal{P}(\mathcal{X}; \Gamma \vdash_\Sigma M : A)$ will take into account exactly the free variables of M, according to the abovementioned intended meaning.

$$\frac{\Gamma \vdash_\Sigma A : K}{\Gamma \vdash_\Sigma A =_\beta A : K}\,(\mathsf{F{\cdot}Refl{\cdot}eq}) \qquad \frac{\Gamma \vdash_\Sigma B =_\beta A : K}{\Gamma \vdash_\Sigma A =_\beta B : K}\,(\mathsf{F{\cdot}Sym{\cdot}eq})$$

$$\frac{\Gamma \vdash_\Sigma A =_\beta B : K \quad \Gamma \vdash_\Sigma B =_\beta C : K}{\Gamma \vdash_\Sigma A =_\beta C : K}\,(\mathsf{F{\cdot}Trans{\cdot}eq}) \qquad \frac{\Gamma \vdash_\Sigma A \mapsto_\beta B : K}{\Gamma \vdash_\Sigma A =_\beta B : K}\,(\mathsf{F{\cdot}Red{\cdot}eq})$$

Fig. 4. LF_K Definitional Equality (Family rules)

$$\mathsf{A}_1 : \phi \to (\psi \to \phi) \qquad \mathsf{K} : \Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$$
$$\mathsf{A}_2 : (\phi \to (\psi \to \xi)) \to (\phi \to \psi) \to (\phi \to \xi) \qquad \mathsf{4} : \Box\phi \to \Box\Box\phi$$
$$\mathsf{A}_3 : (\neg\phi \to \neg\psi) \to ((\neg\phi \to \psi) \to \phi) \qquad \top : \Box\phi \to \phi$$
$$\mathsf{MP} : \frac{\phi \quad \phi \to \psi}{\psi} \qquad \mathsf{NEC} : \frac{\phi}{\Box\phi}$$

Fig. 5. Hilbert style rules for Modal Logic S_4

Moreover, it is worth noticing that, once the "good families" are chosen, predicates are automatically defined as a consequence (look at the examples in the next section).

As far as definitional equality is concerned, due to lack of space, we give in Figure 4 only the rules on families, the ones for kinds and objects being similar. Notice that typing, β-reduction, and equality are mutually defined. Moreover, β-reduction is parametric over a (finite) set of good predicates, that is in LF_K we can combine several good predicates at once.

Finally, notice that our approach is different from static approaches, where "bad" terms are ruled out *a priori* via rigid type disciplines. Namely, in our framework stuck redexes can become enabled in the future. Consider, *e.g.* a redex $(\lambda_{\mathcal{P}_1} x{:}A.M)\ N$ which is stuck because a free variable y occurring into N does not satisfy the constraint imposed by predicate $\mathcal{P}_1$. Then, it could be the case that such redex is inserted into a context where y will be instantiated by a term P, by means of an outer (non-stuck) redex, like, *e.g.* in $(\lambda_{\mathcal{P}_2} y{:}B.(\lambda_{\mathcal{P}_1} x{:}A.M)\ N)\ P$. The resulting redex $(\lambda_{\mathcal{P}_1} x{:}A[P/y].M[P/y])\ N[P/y]$ could then fire since the constraint imposed by the predicate $\mathcal{P}_1$ is satisfied by $N[P/y]$.

3 Instantiating LF_K to Modal Logics

The Conditional Logical Framework is quite expressive. By instantiating the set of predicates, we can recover various known and new interesting Logical Frameworks. The original LF can be recovered by considering the trivial predicate induced by the set $\mathcal{A}$ of all families. More interesting instances of LF_K are introduced below for providing smooth encodings of Modal Logics.

Modal Logic in Hilbert style. The expressive power of the Conditional Logical Framework allows to encode smoothly and uniformly both rules of proof as well as rules of derivation. We recall that the former are rules which apply only to premises which do not depend on any assumption, such as the rule of *necessitation* in Modal Logics, while

the latter apply to all premises, such as *modus ponens*. The idea is to use a conditioned Π-abstraction in rules of proof and a standard Π-abstraction in rules of derivation.

We shall not develop here the encodings of all the gamut of Modal Logics, in Hilbert style, which is extensively treated in [AHMP98]. By way of example, we shall only give the signature for classical S_4 (see Figure 5) in Hilbert style (see Figure 6), which features necessitation (rule NEC in Figure 5) as a rule of proof. For notational convention in Figure 6 and in the rest of this section, we will denote by o^n the expression $\underbrace{o \to o \to \ldots \to o}_{n}$. The target language of the encoding is the instance of LF_κ, obtained by combining standard β-reduction with the β-reduction conditioned by the predicate Closed_o induced by the set $\mathcal{A} = \{o\}$. Intuitively, $\mathsf{Closed}_o(\mathsf{Fv}(M); \Gamma \vdash_{S_4} M : \mathsf{True}(\phi))$ holds iff "all free variables occurring in M belong to a subterm which can be typed in the derivation with o". This is precisely what is needed to encode it correctly, provided o is the type of propositions. Indeed, if all the free variables of a proof term satisfy such condition, it is clear, by inspection of the typing system's object rules (see Figure 1), that there cannot be subterms of type $\mathsf{True}(\ldots)$ containing free variables. Intuitively, this corresponds to the fact that the proof of the encoded modal formula does not depend on any assumptions. The following Adequacy Theorem can be proved in the standard way, using the properties of LF_κ in Section 4.

Theorem 1 (Adequacy of the encoding of S_4 - Syntax)
Let ϵ be an encoding function (induced by the signature in Figure 6) mapping object level formulæ of S_4 into the corresponding canonical terms[1] *of LF_κ of type o. If ϕ is a propositional modal formula with propositional free variables $x_1, \ldots, x_k$, then the following judgment $\Gamma \vdash_{S_4} \epsilon(\phi) : o$ is derivable, where $\Gamma \equiv x_1{:}o, \ldots, x_k{:}o$ and each x_i is a free propositional variable in ϕ. Moreover, if we can derive in LF_κ $\Gamma \vdash_{S_4} M : o$ where $\Gamma \equiv x_1{:}o, \ldots, x_k{:}o$ and M is a canonical form, then there exists a propositional modal formula ϕ with propositional free variables $x_1, \ldots, x_k$ such that $M \equiv \epsilon(\phi)$.*

The proof amounts to a straightforward induction on the structure of ϕ (first part) and on the structure of M (second part). After proving the adequacy of syntax, we can proceed with the more interesting theorems about the adequacy of the truth judgments.

Theorem 2 (Adequacy of the encoding of S_4 - Truth Judgment)
$\phi_1, \ldots, \phi_h \vdash_{S_4} \phi$ if and only if there exists a canonical form M such that

$$\Gamma, y_1{:}\mathsf{True}(\epsilon(\phi_1)), \ldots, y_h{:}\mathsf{True}(\epsilon(\phi_h)) \vdash_{S_4} M : \mathsf{True}(\epsilon(\phi))$$

where $\Gamma \equiv x_1{:}o, \ldots, x_k{:}o$ for each x_i free propositional variable in $\phi_1, \ldots, \phi_h, \phi$.

Classical Modal Logic S_4 and S_5 in Prawitz Style. By varying the notion of good types in the general format of LF_κ, one can immediately generate Logical Frameworks which accommodate both classical Modal Logics S_4 and S_5 in Natural Deduction style introduced by Prawitz. Figure 7 shows common and specific rules of S_4 and S_5.

[1] In this case, as in [HHP93], in stating the adequacy theorem it is sufficient to consider long $\lambda\beta\eta$-normal forms without stuck redexes as canonical forms. Namely, non-reducible terms with stuck redexes must contain free variables not belonging to subterms typable with o, and clearly such terms do not correspond to any S_4-formula.

Propositional Connectives and Judgments

o : Type $\quad \supset : o^3 \quad \neg : o^2 \quad \Box : o^2 \quad \mathsf{True} : o \to \mathsf{Type}$

Propositional Axioms

$\mathsf{A}_1 \quad : \Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi \supset (\psi \supset \phi))$

$\mathsf{A}_2 \quad : \Pi\phi{:}o.\ \Pi\psi{:}o.\ \Pi\xi{:}o.\ \mathsf{True}((\phi \supset (\psi \supset \xi)) \supset (\phi \supset \psi) \supset (\phi \supset \xi))$

$\mathsf{A}_3 \quad : \Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}((\neg\psi \supset \neg\phi) \supset ((\neg\psi \supset \phi) \supset \psi))$

Modal Axioms

$\mathsf{K} \quad : \Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\Box(\phi \supset \psi) \supset (\Box\phi \supset \Box\psi))$

$4 \quad : \Pi\phi{:}o.\ \mathsf{True}(\Box\phi \supset \Box\Box\phi)$

$\top \quad : \Pi\phi{:}o.\ \mathsf{True}(\Box\phi \supset \phi)$

Rules

$\mathsf{MP} \quad : \Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi) \to \mathsf{True}(\phi \supset \psi) \to \mathsf{True}(\psi)$

$\mathsf{NEC} : \Pi\phi{:}o.\ \Pi_{\mathsf{Closed}_o} x{:}\mathsf{True}(\phi).\ \mathsf{True}(\Box\phi)$

Fig. 6. The signature Σ_{S_4} for classic S_4 Modal Logic in Hilbert style

Modal Logic common rules in Natural Deduction style

$$\frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\,(\wedge\mathsf{I}) \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\,(\wedge\mathsf{E}_1) \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\,(\wedge\mathsf{E}_2)$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\,(\vee\mathsf{I}_1) \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}\,(\vee\mathsf{I}_2) \qquad \frac{\Gamma \vdash \phi \vee \psi \quad \Gamma, \phi \vdash \xi \quad \Gamma, \psi \vdash \xi}{\Gamma \vdash \xi}\,(\vee\mathsf{E})$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\,(\to\mathsf{I}) \qquad \frac{\Gamma \vdash \phi \to \psi \quad \Gamma \vdash \phi}{\psi}\,(\to\mathsf{E})$$

$$\frac{\Gamma, \phi \vdash \neg\phi}{\Gamma \vdash \neg\phi}\,(\neg\mathsf{I}) \qquad \frac{\Gamma \vdash \neg\phi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi}\,(\neg\mathsf{E}) \qquad \frac{\Gamma, \neg\phi \vdash \phi}{\Gamma \vdash \phi}\,(\mathsf{RAA})$$

Specific rules for Modal Logic S_4 in Natural Deduction style

$$\frac{\Box\Gamma \vdash \phi}{\Box\Gamma \vdash \Box\phi}\,(\Box\mathsf{I}) \qquad \frac{\Gamma \vdash \Box\phi}{\Gamma \vdash \phi}\,(\Box\mathsf{E})$$

Specific rules for Modal Logic S_5 in Natural Deduction style

$$\frac{\Box\Gamma_0, \neg\Box\Gamma_1 \vdash \phi}{\Box\Gamma_0, \neg\Box\Gamma_1 \vdash \Box\phi}\,(\Box\mathsf{I}) \qquad \frac{\Gamma \vdash \Box\phi}{\Gamma \vdash \phi}\,(\Box\mathsf{E})$$

Fig. 7. Modal Logic (common rules and $S_{4,5}$ rules) in LF_K

We combine again standard β-reduction with a suitable notion of β-reduction conditioned by a predicate Boxed. As in the previous case such predicate can be defined by fixing a suitable notion of good type. In the case of S_4 a type is good if it is of the shape

Propositional Connectives and Judgments

o : Type and : o^3 or : o^3 $\supset$: o^3 $\neg$: o^2 $\Box$: o^2 True : $o \to$ Type

Propositional Rules

$\mathsf{And_I}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi) \to \mathsf{True}(\psi) \to \mathsf{True}(\phi\ \mathsf{and}\ \psi)$

$\mathsf{And_{E_1}}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi\ \mathsf{and}\ \psi) \to \mathsf{True}(\phi)$

$\mathsf{And_{E_2}}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi\ \mathsf{and}\ \psi) \to \mathsf{True}(\psi)$

$\mathsf{Or_{I_1}}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi) \to \mathsf{True}(\phi\ \mathsf{or}\ \psi)$

$\mathsf{Or_{I_2}}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\psi) \to \mathsf{True}(\phi\ \mathsf{or}\ \psi)$

$\mathsf{Or_E}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi\ \mathsf{or}\ \psi) \to (\mathsf{True}(\phi) \to \mathsf{True}(\xi)) \to (\mathsf{True}(\psi) \to \mathsf{True}(\xi)) \to \mathsf{True}(\xi)$

$\mathsf{Imp_I}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ (\mathsf{True}(\phi) \to \mathsf{True}(\psi)) \to \mathsf{True}(\phi \supset \psi)$

$\mathsf{Imp_E}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\phi \supset \psi) \to \mathsf{True}(\phi) \to \mathsf{True}(\psi)$

$\mathsf{Neg_I}$: $\Pi\phi{:}o.\ (\mathsf{True}(\phi) \to \mathsf{True}(\neg\phi)) \to \mathsf{True}(\neg\phi)$

$\mathsf{Neg_E}$: $\Pi\phi{:}o.\ \Pi\psi{:}o.\ \mathsf{True}(\neg\phi) \to \mathsf{True}(\phi) \to \mathsf{True}(\psi)$

RAA : $\Pi\phi{:}o.\ (\mathsf{True}(\neg\phi) \to \mathsf{True}(\phi)) \to \mathsf{True}(\phi)$

Modal Rules

$\mathsf{Box_I}$: $\Pi\phi{:}o.\ \Pi_{\mathsf{Boxed}}x{:}\mathsf{True}(\phi).\ \mathsf{True}(\Box\phi)$

$\mathsf{Box_E}$: $\Pi\phi{:}o.\ \Pi x{:}\mathsf{True}(\Box\phi).\ \mathsf{True}(\phi)$

Fig. 8. The signature Σ_S for classic S_4 or S_5 Modal Logic in Natural Deduction style

$\mathsf{True}(\Box\mathsf{A})$ for a suitable A or o. In the case of S_5 a type is good if it is either of the shape $\mathsf{True}(\Box\mathsf{A})$ or $\mathsf{True}(\neg\Box\mathsf{A})$ or o. Again the intended meaning is that all occurrences of free variables appear in subterms having a $\Box$-type or within a syntactic type o in the case of S_4, and a $\Box$-type or $\neg\Box$-type or within a syntactic type o in the case of S_5.

Thus, *e.g.* for S_4, the encoding of the Natural Deduction ($\Box$I) rule of Prawitz (see Figure 7) can be rendered as $\Pi\phi{:}o.\ \Pi_{\mathsf{Boxed}}x{:}\mathsf{True}(\phi).\ \mathsf{True}(\Box\phi)$, where o:Type represents formulæ, while $\mathsf{True}{:}o \to \mathsf{Type}$ and $\Box{:}o \to o$.

Quite a remarkable property of this signature is that it encodes a slightly more usable version of Natural Deduction S_4 than the one originally introduced by Prawitz. Our formulation is precisely what is needed to achieve a normalization result in the logic which could not be done in the original system of Prawitz. Being able to refer to *boxed subterms*, rather than just boxed variables, is what *makes the difference*. Once again $\mathsf{LF_K}$ encodings *improve presentations of logical systems*!

4 Properties of $\mathsf{LF_K}$

In this section, we study relevant properties of $\mathsf{LF_K}$. We show that, without any extra assumption on the predicates, the type system satisfies a list of basic properties, including the subderivation property, subject reduction and strong normalization. The latter follows easily from the strong normalization result for LF, see [HHP93]. Confluence and judgment decidability can be proved under the assumption that the various predicate reductions nicely combine, in the sense that no reduction can prevent a redex,

which could fire, from firing after the reduction. The difficulty in proving subject reduction and confluence for LF_κ lies in the fact that predicate β-reductions do not have corresponding untyped reductions, while standard proofs of subject reduction and confluence for dependent type systems are based on underlying untyped β-reductions (see *e.g.* [HHP93]). We provide an original technique, based solely on typed β-reductions, providing a fine analysis of the structure of terms which are β-equivalent to Π-terms.

In the following, we will denote by $\Gamma \vdash_\Sigma \alpha$ any judgment defined in LF_κ. The proof of the following theorem is straightforward.

Theorem 3 (Basic Properties)

Subderivation Property

1. *Any derivation of $\Gamma \vdash_\Sigma \alpha$ has subderivations of Σ sig and $\vdash_\Sigma \Gamma$.*
2. *Any derivation of $\Sigma, a{:}K$ sig has subderivations of Σ sig and $\vdash_\Sigma K$.*
3. *Any derivation of $\Sigma, f{:}A$ sig has subderivations of Σ sig and $\vdash_\Sigma A : \mathsf{Type}$.*
4. *Any derivation of $\vdash_\Sigma \Gamma, x{:}A$ has subderivations of Σ sig and $\Gamma \vdash_\Sigma A : \mathsf{Type}$.*
5. *Given a derivation of $\Gamma \vdash_\Sigma \alpha$ and any subterm occurring in the subject of the judgment, there exists a derivation of a smaller length of a judgment having that subterm as a subject.*
6. *If $\Gamma \vdash_\Sigma A : K$, then $\Gamma \vdash_\Sigma K$.*
7. *If $\Gamma \vdash_\Sigma M : A$, then $\Gamma \vdash_\Sigma A : \mathsf{Type}$ if there are no stuck redexes in A.*

Derivability of Weakening and Permutation

If Γ and Δ are valid contexts, and every declaration occurring in Γ also occurs in Δ, then $\Gamma \vdash_\Sigma \alpha$ implies $\Delta \vdash_\Sigma \alpha$.

Transitivity

If $\Gamma, x{:}A, \Delta \vdash_\Sigma \alpha$ and $\Gamma \vdash_\Sigma M : A$, then $\Gamma, \Delta[M/x] \vdash_\Sigma \alpha[M/x]$.

Convertibility of types in domains

1. *For all $\Gamma, x{:}A, \Delta \vdash_\Sigma \alpha$ and $\Gamma, \Delta \vdash_\Sigma A =_\beta A' : K$, then $\Gamma, x{:}A', \Delta \vdash_\Sigma \alpha$.*
2. *If $\mathcal{P}(\mathcal{X}; \Gamma, x{:}A, \Delta \vdash_\Sigma M : B)$ holds and $\Gamma, \Delta \vdash_\Sigma A =_\beta A' : K$, then $\mathcal{P}(\mathcal{X}; \Gamma, x{:}A', \Delta \vdash_\Sigma M : B)$ holds.*

Strong normalization of LF_κ follows from the one of LF, since there is a trivial map of LF_κ in LF, which simply forgets about predicates. Thus, if there would be an infinite reduction in LF_κ, this would be mapped into an infinite reduction in LF.

Theorem 4 (Strong Normalization)

1. *If $\Gamma \vdash_\Sigma K$, then $K \in \mathsf{SN}^{\mathcal{K}}$.*
2. *if $\Gamma \vdash_\Sigma A : K$, then $A \in \mathsf{SN}^{\mathcal{F}}$.*
3. *if $\Gamma \vdash_\Sigma M : A$, then $M \in \mathsf{SN}^{\mathcal{O}}$.*

Where $\mathsf{SN}^{\{\mathcal{K},\mathcal{F},\mathcal{O}\}}$ denotes the set of strongly normalizing terms of kinds, families, and objects, respectively.

In the following we will denote by $\Gamma \vdash_\Sigma A \rightleftharpoons_\beta B : K$ the fact that either $\Gamma \vdash_\Sigma A \mapsto_\beta B : K$ or $\Gamma \vdash_\Sigma B \mapsto_\beta A : K$ holds. Moreover, in the next results we will use a *measure* of the complexity of the proofs of judgments which takes into account all the rules applied in the derivation tree. More precisely, we have the following definition:

Definition 3 (Measure of a derivation)
Given a proof $\mathcal{D}$ of the judgment $\Gamma \vdash_\Sigma \alpha$, we define the measure of $\mathcal{D}$, denoted by $\#\mathcal{D}$, as the number of all the rules applied in the derivation of $\mathcal{D}$ itself.

The following lemma is easily proved by induction on $\#\mathcal{D}$.

Lemma 2 (Reduction/Expansion).
For any derivation $\mathcal{D} : \Gamma \vdash_\Sigma A =_\beta B : K$, either $A \equiv B$ or there exist $C_1, \ldots, C_n$ ($n \geq 0$) such that:

1. *There exist $\mathcal{D}_1 : \Gamma \vdash_\Sigma A \rightleftharpoons_\beta C_1 : K$ and $\mathcal{D}_2 : \Gamma \vdash_\Sigma C_1 \rightleftharpoons_\beta C_2 : K \ldots$ and $\mathcal{D}_n : \Gamma \vdash_\Sigma C_{n-1} \rightleftharpoons_\beta C_n : K$ and $\mathcal{D}_{n+1} : \Gamma \vdash_\Sigma C_n \rightleftharpoons_\beta B : K$ and, for all $1 \leq i \leq n+1$, we have $\#\mathcal{D}_i < \#\mathcal{D}$.*
2. *For any $1 \leq i \leq n$, we have that there exist $\mathcal{D}'_1 : \Gamma \vdash_\Sigma A =_\beta C_i : K$ and $\mathcal{D}'_2 : \Gamma \vdash_\Sigma C_i =_\beta B : K$ and $\#\mathcal{D}'_1, \#\mathcal{D}'_2 < \#\mathcal{D}$.*

This lemma allows us to recover the structure of a term which is β-equivalent to a Π-term. The proof proceeds by induction on $\#\mathcal{D}$.

Lemma 3 (Key lemma).

1. *If $\mathcal{D} : \Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.K =_\beta K'$ holds, then either $\Pi_\mathcal{P} x{:}A.K \equiv K'$ or there are $\mathcal{P}_1, ..., \mathcal{P}_n$, and $D_1, ..., D_n$, and $M_1, ..., M_n$ ($n \geq 0$), and K_A, $\mathcal{D}_1$, $\mathcal{D}_2$ such that:*
   (a) $K' \equiv ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.(\Pi_\mathcal{P} x{:}A'.K'')) M_n) \ldots) M_1)$.
   (b) $\mathcal{D}_1 : \Gamma \vdash_\Sigma A =_\beta ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.A') M_n) \ldots) M_1) : K_A$.
   (c) $\mathcal{D}_2 : \Gamma, x{:}A \vdash_\Sigma K =_\beta ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.K'') M_n) \ldots) M_1)$.
   (d) $\#\mathcal{D}_1, \#\mathcal{D}_2 < \#\mathcal{D}$.
2. *If $\mathcal{D} : \Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.B =_\beta C : K$ holds, then either $\Pi_\mathcal{P} x{:}A.B \equiv C$ or there are $\mathcal{P}_1, ..., \mathcal{P}_n$, and $D_1, ..., D_n$, and $M_1, ..., M_n$ ($n \geq 0$), and K_A, K_B, and $\mathcal{D}_1$, $\mathcal{D}_2$ such that:*
   (a) $C \equiv ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.(\Pi_\mathcal{P} x{:}A'.B')) M_n) \ldots) M_1)$.
   (b) $\mathcal{D}_1 : \Gamma \vdash_\Sigma A =_\beta ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.A') M_n) \ldots) M_1) : K_A$.
   (c) $\mathcal{D}_2 : \Gamma, x{:}A \vdash_\Sigma B =_\beta ((\lambda_{\mathcal{P}_1} y_1{:}D_1. \ldots ((\lambda_{\mathcal{P}_n} y_n{:}D_n.B') M_n) \ldots) M_1) : K_B$.
   (d) $\#\mathcal{D}_1, \#\mathcal{D}_2 < \#\mathcal{D}$.

Corollary 1 (Π's injectivity).

1. *If $\Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.K =_\beta \Pi_\mathcal{P} x{:}A'.K'$, then $\Gamma \vdash_\Sigma A =_\beta A' : K_A$ and $\Gamma, x{:}A \vdash_\Sigma K =_\beta K'$.*
2. *If $\Gamma \vdash_\Sigma \Pi_\mathcal{P} x{:}A.B =_\beta \Pi_\mathcal{P} x{:}A'.B' : K$, then $\Gamma \vdash_\Sigma A =_\beta A' : K'$ and $\Gamma, x{:}A \vdash_\Sigma B =_\beta B' : K''$.*

The proof of the following theorem uses the Key Lemma.

Theorem 5 (Unicity, Abstraction and Subject Reduction)

Unicity of Types and Kinds

1. *If $\Gamma \vdash_\Sigma A : K_1$ and $\Gamma \vdash_\Sigma A : K_2$, then $\Gamma \vdash_\Sigma K_1 =_\beta K_2$.*
2. *If $\Gamma \vdash_\Sigma M : A_1$ and $\Gamma \vdash_\Sigma M : A_2$, then $\Gamma \vdash_\Sigma A_1 =_\beta A_2 : K$.*

Abstraction Typing

1. *If $\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.T : \Pi_\mathcal{P} x{:}A'.T'$, then $\Gamma \vdash_\Sigma A =_\beta A' : K$.*

2. *If* $\Gamma \vdash_\Sigma \lambda_\mathcal{P} x{:}A.T : \Pi_\mathcal{P} x{:}A.T'$*, then* $\Gamma, x{:}A \vdash_\Sigma T : T'$.

Subject Reduction

1. *If* $\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.K)\, N$*, then* $\Gamma \vdash_\Sigma K[N/x]$.
2. *If* $\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.B)\, N : K$ *and* $\mathcal{P}(\mathsf{Fv}(N); \Gamma \vdash_\Sigma N : A)$ *holds, then* $\Gamma \vdash_\Sigma B[N/x] : K$.
3. *If* $\Gamma \vdash_\Sigma (\lambda_\mathcal{P} x{:}A.M)\, N : C$ *and* $\mathcal{P}(\mathsf{Fv}(N); \Gamma \vdash_\Sigma N : A)$ *holds, then* $\Gamma \vdash_\Sigma M[N/x] : C$.

In the following, we consider notions of reduction for LF_κ that are *well-behaved* in the following sense:

1. a redex which can fire, can still fire after any β-reduction in its argument (possibly corresponding to a different predicate);
2. a redex which can fire, can still fire after application to its argument of a substitution coming from another reduction.

Formally:

Definition 4 (Well behaved β-reduction)
Assume that the LF_κ *β-reduction is determined by the set* $\mathbf{P}$ *of good predicates. Then the β-reduction is* well-behaved *if, for all* $\mathcal{P}, \mathcal{P}' \in \mathbf{P}$*, the following two conditions are satisfied:*

1. *If* $\mathcal{P}(\mathsf{Fv}(N); \Gamma \vdash_\Sigma N : A)$ *holds and* $\Gamma \vdash_\Sigma N \mapsto_\beta N' : A$*, then* $\mathcal{P}(\mathsf{Fv}(N'); \Gamma \vdash_\Sigma N' : A)$ *holds.*
2. *If* $\mathcal{P}(\mathsf{Fv}(N); \Gamma', y{:}A'; \Gamma \vdash_\Sigma N : A)$ *and* $\mathcal{P}'(\mathsf{Fv}(N'); \Gamma' \vdash_\Sigma N' : A')$ *hold, then* $\mathcal{P}(\mathsf{Fv}(N[N'/y]); \Gamma', \Gamma[N'/y] \vdash_\Sigma N[N'/y] : A[N'/y])$ *holds.*

Definition 4 above allows one to combine several notions of predicate reduction, provided the latter are all well-behaved.

Since LF_κ is strongly normalizing, in order to prove confluence of the system, by *Newman's Lemma*, it is sufficient to show that LF_κ β-reduction is *locally confluent*, *i.e.* (in the case of objects) if $\Gamma \vdash_\Sigma M_1 \mapsto_\beta M_2 : C$ and $\Gamma \vdash_\Sigma M_1 \mapsto_\beta M_3 : C$, then there exists M_4 such that $\Gamma \vdash_\Sigma M_2 \mapsto\!\!\!\!\to_\beta M_4 : C$ and $\Gamma \vdash_\Sigma M_3 \mapsto\!\!\!\!\to_\beta M_4 : C$. Under the hypothesis that β-reduction is well-behaved, using Theorem 5, we can prove that the reduction is locally confluent.

Theorem 6 (Local Confluence)
If β-reduction is well behaved, then it is locally confluent.

Finally, from Newman's Lemma, using Theorems 4 and 6, we have:

Theorem 7 (Confluence)
Assume β-reduction is well behaved. Then the relation $\mapsto_\beta$ is confluent, i.e.:

1. *If* $\Gamma \vdash_\Sigma K_1 \mapsto\!\!\!\!\to_\beta K_2$ *and* $\Gamma \vdash_\Sigma K_1 \mapsto\!\!\!\!\to_\beta K_3$*, then there exists* K_4 *such that* $\Gamma \vdash_\Sigma K_2 \mapsto\!\!\!\!\to_\beta K_4$ *and* $\Gamma \vdash_\Sigma K_3 \mapsto\!\!\!\!\to_\beta K_4$.
2. *If* $\Gamma \vdash_\Sigma A_1 \mapsto\!\!\!\!\to_\beta A_2 : K$ *and* $\Gamma \vdash_\Sigma A_1 \mapsto\!\!\!\!\to_\beta A_3 : K$*, then there exists* A_4 *such that* $\Gamma \vdash_\Sigma A_2 \mapsto\!\!\!\!\to_\beta A_4 : K$ *and* $\Gamma \vdash_\Sigma A_3 \mapsto\!\!\!\!\to_\beta A_4 : K$.

3. *If* $\Gamma \vdash_\Sigma M_1 \mapsto\!\!\!\!\rightarrow_\beta M_2 : C$ *and* $\Gamma \vdash_\Sigma M_1 \mapsto\!\!\!\!\rightarrow_\beta M_3 : C$*, then there exists* M_4 *such that* $\Gamma \vdash_\Sigma M_2 \mapsto\!\!\!\!\rightarrow_\beta M_4 : C$ *and* $\Gamma \vdash_\Sigma M_3 \mapsto\!\!\!\!\rightarrow_\beta M_4 : C$.

Judgements decidability show that LF_κ can be used as a framework for proof checking.

Theorem 8 (Judgements decidability of LF_κ)
If $\mapsto_\beta$ *is well-behaved, then it is decidable whether* $\Gamma \vdash_\Sigma \alpha$ *is derivable.*

The standard pattern of the proof applies, provided we take care that reductions are typed in computing the normal form of a type.

It is easy to show that, for all instances of LF_κ considered in Section 3, the corresponding β-reductions are well behaved, thus judgement decidability holds.

5 Conclusions and Directions for Future Work

In this paper, we have investigated the language theory of the Conditional Logical Framework LF_κ, which subsumes the Logical Framework LF of [HHP93], and generates new Logical Frameworks. These can feature a very broad spectrum of generalized typed (possibly by value) β-reductions, together with an expressive type system which records when such reductions have not yet fired. The key ingredient in the typing system is a decomposition of the standard term-application rule. A very interesting feature of our system is that it allows for dealing with values induced by the typing system, *i.e.* values which are determined by the typing system, through the notion of good predicates. We feel that our investigation of LF_κ is quite satisfactory: we have proved major metatheoretical results, such as strong normalization, subject reduction and confluence (this latter under a suitable assumption). For LF_κ we have achieved decidability, which legitimates it as a metalanguage for proof checking and interactive proof editing. We have shown how suitable instances of LF_κ provide smooth encodings of Modal Logics, compared with the heavy machinery needed when we work directly into LF, see *e.g.* [AHMP98]. Namely, the work of specifying the variable occurrence side-conditions is factored out once and for all into the framework.

Here is a list of comments and directions for future work.

- Some future efforts should be devoted to the task of investigating the structure of canonical forms including stuck redexes. Such analysis could clarify the rôle of stuck β-reductions and stuck terms in the activity of encoding object logics into LF_κ. Moreover, following the approach carried out in [WCPW02], we could benefit from a presentation of LF_κ based upon a clear characterization of canonical forms in order to avoid the notion of β-conversion and the related issues.
- We believe that our metalogical Framework has some considerable potential. In particular, it could be useful for modeling dynamic situations, where the static approach of rigid typed disciplines is not sufficient. We plan to carry out more experiments in the future, *e.g.* in the field of reactive systems, where the rôle of stuck redexes could be very helpful in modeling the dynamics of variables instantiations.
- Our results should scale up to systems corresponding to the full Calculus of Constructions [CH88].
- Is there an interesting Curry-Howard isomorphism for LF_κ, and for other systems blending rewriting facilities and higher order calculi?

– Investigate whether LF_κ could give sharp encodings of Relevance and Linear Logics. Is the notion of good predicate involved in the definition of LF_κ useful in this respect? Or do we need a different one?
– Compare with work on Deduction Modulo [DHK03].
– In [KKR90], Kirchner-Kirchner-Rusinowitch developed an *Algebraic Logical Framework* for first-order constrained deduction. Deduction rules and constraints are given for a first-order logic with equality. Enhancing LF_κ with constraints seems to be a perfect fit for a new race of metalanguages for proof checking and automatic theorem proving. Without going much into the details of our future research, the abstraction-term could, indeed, have the shape $\lambda_{\mathcal{P}}\overline{x};\mathcal{C}.M$, where $\mathcal{P}$ records the first-order formula, $\overline{x}$ is a vector of variables occurring in the formula and $\mathcal{C}$ are constraints over $\overline{x}$.
– Until now, the predicate states a condition that takes as *input* the argument and its type. It would be interesting to extend the framework with another predicate, say $\mathcal{Q}$, applied to the body of the function. The abstraction would then have the form $\lambda_{\mathcal{P}}x{:}A.M^{\mathcal{Q}}$. This extension would put conditions on the function *output*, so leading naturally to a framework for defining Program Logics *à la* Hoare-Floyd.
– Implement new proof assistants based on dependent type systems, like *e.g.* Coq, based on LF_κ.

References

[AHMP98] A. Avron, F. Honsell, M. Miculan, and C. Paravano. Encoding Modal Logics in Logical Frameworks. *Studia Logica*, 60(1):161–208, 1998.

[BCKL03] G. Barthe, H. Cirstea, C. Kirchner, and L. Liquori. Pure Pattern Type Systems. In *Proc. of POPL*, pages 250–261. The ACM Press, 2003.

[CH84] M. Cresswell and G. Hughes. *A companion to Modal Logic*. Methuen, 1984.

[CH88] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2/3):95–120, 1988.

[CKL01a] H. Cirstea, C. Kirchner, and L. Liquori. Matching Power. In *Proc. of RTA*, volume 2051 of *Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2001.

[CKL01b] H. Cirstea, C. Kirchner, and L. Liquori. The Rho Cube. In *Proc. of FOSSACS*, volume 2030 of *Lecture Notes in Computer Science*, pages 166–180, 2001.

[DHK03] G. Dowek, T. Hardin, and C. Kirchner. Theorem Proving Modulo. *Journal of Automated Reasoning*, 31(1):33–72, 2003.

[HHP93] R. Harper, F. Honsell, and G. Plotkin. A Framework for Defining Logics. *Journal of the ACM*, 40(1):143–184, 1993. Preliminary version in proc. of LICS'87.

[HLL07] F. Honsell, M. Lenisa, and L. Liquori. A Framework for Defining Logical Frameworks. *Computation, Meaning and Logic. Articles dedicated to Gordon Plotkin, Electr. Notes Theor. Comput. Sci.*, 172:399–436, 2007.

[KKR90] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with Symbolic Constraints. Technical Report 1358, INRIA, Unité de recherche de Lorraine, Vandoeuvre-lès-Nancy, FRANCE, 1990.

[NPP08] A. Nanevski, F. Pfenning, and B. Pientka. Contextual Model Type Theory. *ACM Transactions on Computational Logic*, 9(3), 2008.

[WCPW02] K. Watkins, I. Cervesato, F. Pfenning, and D. Walker. A Concurrent Logical Framework I: Judgments and Properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002.